Bill J. Ellis
February 1, 2005

Six Months

Contents

[*]Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe ϵ-baked ideas, where $1 \geq \epsilon \geq 0$. (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

1 Introduction

1.1 Previous Project

Title	**Automatic Guidance for the Formal Verification of High Integrity Ada**
Grant	GR/R24081 - Part of the EPSRC Critical Systems research programme
Collaboration	Praxis High Integrity Systems
Based at	Heriot-Watt University
Duration	01/09/2001 to 31/08/04
This project was fundamentally a research project. The project focused on the development of the NuSPADE system. NuSPADE demonstrated an integrated approach to automated program reasoning. In particular, NuSPADE focused on proving the absence of exception freedom within the SPARK programming language.	

1.2 This Project

Title	**Toward Increased Verification Automation for High Integrity Software Engineering**
Grant	GR/T12289/01 - EPSRC Collaborative Training Account - Research Assistant Industrial Secondment (RAIS) Grant
Collaboration	Praxis High Integrity Systems
Based at	Praxis High Integrity Systems
Duration	17/01/05 to 17/07/05

This project follows in the footsteps of the earlier project. However, this project is fundamentally a training project. The aim is to demonstrate the ideas developed in the earlier project to an industrial audience. This will be achieved by implementing a refined, industrial driven, version of NuSPADE. The pursuit of this task will inevitably require training in industrial practise and techniques. Note that, given the limited available time, it is quite unrealistic to develop a genuine commercial version of NuSPADE. Rather our intention is to demonstrate the following hypotheses:

1. A commercial version of NuSPADE could exist
2. It could be cleanly integrated within the SPARK Approach
3. It would automatically provide enhanced program verification support

There is a strong argument that, through NuSPADE, these hypotheses have already been demonstrated in theory. Our intention is simply to modify NuSPADE to demonstrate these hypotheses in practise.
To minimise confusion a new name will be used to refer to the modified version of NuSPADE (the SPADE Helper). The choice of "Helper" indicates that the tool is not solely concerned with proof nor program analysis, rather it offers help in the general task of program reasoning. The two key components of the SPADE Helper will be a modified version of the CLAM Proof Planner (Helper-PP) and a modified version of PropGen (Helper-PG).

2 Plan

2.1 Project Plan

The duration of the project is fixed at 6 months (26 weeks). Subtracting holidays, this leaves 23 working weeks.

WP	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	•																						
1	2		•																					
1	3			•																				
2	1				•	•																		
2	2						•	•	•	•														
2	3										•													
2	4											•	•											
3	1													•	•	•								
4	1																•	•	•	•				
4	2																				•	•	•	•

2.2 Work Package 1

Adapt SPADE proof tools for integration
The central component of our integrated approach is the SPADE Helper. The SPADE Helper loads the verification problem then searches for a proof or offer user guidance accordingly. Fundamentally, the SPADE Helper must interact with the existing SPADE proof tools to retrieve the verification problem. This work package involves deciding on the exact nature of the integration with the SPADE proof tools. Further, some small but essential changes will be required to the SPADE proof tools to support this integration.
Tasks: • **Integration architecture (1 week)** • **Modify SPADE Simplifier (1 week)** • **Modify SPADE Proof Checker (1 week)**

2.2.1 Work Package 1 Tasks

Integration architecture (1 week)
The SPADE Helper will need to be integrated with the two SPADE proof tools: the SPADE Simplifier and the SPADE Proof Checker. Key to the success of the project is ensuring that the SPADE Helper presents a seamless extension to the SPARK tools. Note that it is unlikely that this can be achieved without making some minor adjustments to the SPADE tools.
Deliverables: • **Note: Architecture -** A short note describing the desired architecture and, consequently, the desired behaviour of the SPADE Simplifier and the SPADE Proof Checker.

Modify SPADE Simplifier (1 week)

The SPADE Helper will tackle the verification conditions not proven automatically by the SPADE Simplifier. It is important that the structure of these verification conditions are preserved. This is currently not supported by the Simplifier. Further, the Simplifier may need to be tweaked to support the desired architecture.

NuSPADE does not directly support this. Instead NuSPADE is manually guided toward the verification conditions that the Simplifier does not automatically prove. An old system (called Friendly) described and implemented a suitable extension to the Simplifier. However, Friendly was written to also collect the SPADE Simplifier intermediate hypotheses. In practise these were difficult to collect and rarely useful. Thus the plan here is to extract the core behaviour of the Friendly system, being mindful of the integration architecture.

Deliverables:

- **Note: Describe changes -** A short note describing the changes made to the SPADE Simplifier.
- **System: Modified SPADE Simplifier -** The modified version of the SPADE Simplifier.

Modify SPADE Proof Checker (1 week)

The SPADE Proof Checker is automatically controlled by the SPADE Helper to demonstrate the correctness of planned verification conditions. Currently the SPADE Proof Checker performs various simplifications automatically, making this task very difficult. Some minor tweaks to the SPADE Proof Checker are required. These tweaks are well understood and have already been implemented for NuSPADE. These changes can be copied and documented accordingly.

There are also problems in making the SPADE Proof Checker more amenable to the overall process of automatically checking discovered proof plans. Ideally it should be possible to automatically invoke the SPADE Proof Checker, execute a command log, and have the success of this proof recorded such that POGS now considers the conclusion discharged. Solving this should also address the command log misalignment problem that arises where planning multiple conclusions from the same verification condition.

Note that an early version of the NuSPADE architecture had the core of NuSPADE directly embedded inside the SPADE Proof Checker program. This supported reuse of the SPADE loading mechanisms. To ease the development of this embedded system the SPADE Proof Checker was ported to Sicstus Prolog. Although this embedded approach was later abandoned, a Sicstus port of the SPADE Proof Checker was essentially completed. It would be possible to consider the restoration of the port at this phase. Note that, since the SPADE Helper will treat the SPADE Proof Checker as a black box, such a port is not technically required.

Deliverables:

- **Note: Describe changes -** A short note describing the changes made to the SPADE Proof Checker.
- **System: Modified SPADE Proof Checker -** The modified version of the SPADE Proof Checker.

2.3 Work Package 2

Rationally reconstruct the proof planner
It is fairly straight forward to envisage the ideal behaviour of a proof planner. Edinburgh University have recently decided to focus their proof planning research on the IsaPlanner system - a “generic framework for proof planning in the interactive theorem prover Isabelle”. Unfortunately, the time for this project is very short, we have limited experience with IsaPlanner or Isabelle, and IsaPlanner (though very promising) is still in active development. Thus, while not an ideal proof planner, we intend to use a modified version of the proof planner used in NuSPADE. This work package involves rationally reconstructing NuSPADE to support the integration architecture established in Work Package 1.
Tasks: • **Import verification conditions (2 weeks)** • **Data management enhancements (4 weeks)** • **Testing and experimentation (1 week)** • **Integration with the SPADE Proof Checker (2 weeks)**

2.3.1 Work Package 2 Tasks

Import verification conditions (2 weeks)
Helper-PP will need to access the verification conditions not automatically proved by the SPADE Simplifier. This problem can be decomposed to requiring access to: • **Verification conditions -** The verification conditions themselves. • **Functional Description Language (FDL) -** The additional type based information associated with the verification conditions. • **Rule files -** The rules associated with each SPARK project, including those built-in and those generated by the SPADE Simplifier. For completeness and further experimentation it would be ideal to also support access to user created rule files. Although all of these facilities are available in NuSPADE, they are currently implemented as separate programs executed in batch mode. It will be necessary to modify these programs to support the integration architecture of work package 1.
Deliverables: • **System: Helper-PP -** The modified version of the NuSPADE proof planner, capable of loading verification conditions in line with the integration architecture.

Data management enhancements (4 weeks)

A major weakness in the implementation of NuSPADE is its limited data management facilities. NuSPADE is manually configured to initialise and explore a search space in tune with the targeted problem. In theory this is quite acceptable (and common practise) as proof planning with rippling will terminate. The key issue is finding a proof, not finding a proof quickly. However, in practise, such manual configuration and weak data management choices will adversely affect the usability of Helper-PP. This needs to be addressed.

Identifying all sources of this data management problem will be one of the tasks to be undertaken here. The obvious problems to be addressed are:

- **Eliminate needs file -** The needs file selects the rules to use for a particular problem. This needs file must be replaced with an exhaustive search or some automatically configured constrained search.
- **Wave rule (and regular rule) management -** Currently NuSPADE loads rules and generates wave rules dynamically. This has the potential to be very expensive if a large number of rules is under consideration. A large number of rules is more likely where losing the tightly constrained needs file. It is likely this problem will be resolved through the introduction of rule caching.
- **Method and critic loading -** Methods and critics must be loaded prior to planning. In NuSPADE this process is tightly coupled with the existing rule management. Further, the ordering of this process directly affects the search strategy, radically affecting the behaviour of the planner. It is probably necessarily (and certainly desirable) to refine the existing method and critics loading process in light of the above changes.
- **Core planner -** Following the above changes it is expected that tweaks will be required to the core planning engine. Fortunately, although NuSPADE is a relatively large system, the core planning engine is quite small.

Note that the aim here is to give Helper-PP a sound data management infrastructure, not to configure this infrastructure for optional performance.

Deliverables:

- **Note: Describe changes -** Describes the rational and nature of the changes made to NuSPADE. This should be a short, high-level discussion.
- **System: Helper-PP -** A further refined version of the NuSPADE proof planner, with improved data management facilities.

Testing and experimentation (1 week)

Once Helper-PP is operating it may be tested on a collection of previously considered examples. The primary intention here is to explore the planning capability of Helper-PP in comparison to NuSPADE.

Deliverables:

- **Data: Examples -** A collection of examples that successfully produce fully instantiated proof plans.
- **Note: Observations -** A note listing the interesting observable features of Helper-PP. In particular, problems and issues encountered will be documented.

Integration with the SPADE Proof Checker (2 weeks)

In work package 1 the SPADE Proof Checker is configured for integration with the SPADE Helper. Following the earlier tasks in this work package, the SPADE Helper will be able to plan some examples. Here these plans are extracted as command logs and executed within the SPADE Proof Checker.

NuSPADE supports the extraction of a SPADE Proof Checker command log via a utility program called Digger. Digger takes as input a discovered proof plan and outputs the corresponding SPADE Proof Checker command log. Note that, since the SPADE Proof Checker is not a true tactic based theorem prover, the data stored in the proof plan must be more detailed than usual to support this process. The coupling between NuSPADE and the SPADE Proof Checker is weak. The user has to manually invoke the SPADE Proof Checker and execute the relevant script. This will need to be addressed. Further, given the tight coupling between planning and Digger it is likely some small tweaks will be required to Digger for its incorporation in the SPADE Helper.

Deliverables:

- **System: Helper-PP -** A further refined version of the NuSPADE proof planner, supporting the generation and execution of a SPADE Proof Checker command log.

2.4 Work Package 3

Investigate coupling to program analysis

NuSPADE supports program analysis for appropriately configured examples via the PropGen system. Unfortunately a realistic program analysis system would be a major engineering task. Thus PropGen takes various short cuts, focusing on our area of research. Nevertheless, it would be interesting for the purposes of demonstration to incorporate this system with the SPADE Helper. The aim here is to focus on the interface, giving a sample of how a full implementation might behave.

This work package involves integrating a modified version of PropGen with the SPADE Helper.

Tasks:

- **Integrate the SPADE Helper with modified PropGen (3 weeks)**

2.4.1 Work Package 3 Tasks

Integrate the SPADE Helper with modified PropGen (3 weeks)

In our integrated approach the proof planner is able to request the services of a program analysis oracle during proof planning. Such an interface is directly supported by the proof planning critics mechanism. However, thus far, this coupling has been simulated rather than implemented.

Further, PropGen has a very limited interface. In particular, PropGen does not filter and present properties. Although this is a fairly trivial task, it will involve some symbolic manipulation. Addressing the PropGen interface will be important in demonstrating the seamless discovery of program properties.

Note that there is the potential to explore a more industrially viable form of program analysis. For example, the SPARK Examiner may be modified to output program details and a suitable program created to analyse these details. However, given the limited time, it is suspected it would be difficult to deliver on this. Nevertheless, it may be interesting to explore the practicality of such an exercise, making notes about how this may be tackled.

Deliverables:

- **System: Helper-PP -** A further refined version of the NuSPADE proof planner, now integrated with a modified version of PropGen.
- **Data: Examples -** A collection of sample programs that are analysable by the SPADE Helper and involve program analysis.

2.5 Work Package 4

Evaluation and publication
This work package involves the evaluation and refinement of the SPADE Helper in view of publishing an academic paper.
Tasks: • **Evaluate and refine the SPADE Helper (4 weeks)** • **Write paper (4 weeks)**

2.5.1 Work Package 4 Tasks

Evaluate and refine the SPADE Helper (4 weeks)
The SPADE Helper will be evaluated on various examples. In light of this evaluation there will inevitably be aspects warranting refinement. The refinements will be targeted to support the goal of writing an academic paper. It is likely that this will include refining existing and developing new methods and critics.
Deliverables: • **Data: Results -** The examples considered during the evaluation and their results. • **System: Helper-PP -** The refined version of the SPADE Helper.

Write paper (4 weeks)
Following the evaluation of the SPADE Helper an academic paper will be written. This will likely involve introspecting on aspects from all of the above work packages.
Deliverables: • **Note: Paper -** An academic paper ready for submission to a suitable venue.

3 Conclusions

The project plan is fairly well constrained, involving various extensions and modifications to existing tools. Given the short duration of the project, and the emphasis on industrial applicability, the raw results are unlikely to improve upon those for NuSPADE. Nevertheless, the development of an industrial driven, proof planning based approach to program verification should be interesting to both academic and industry audiences.